RFC: 761
IEN: 129

DOD STANDARD

TRANSMISSION CONTROL PROTOCOL

January 1980

prepared for

Defense Advanced Research Projects Agency
Information Processing Techniques Office
1400 Wilson Boulevard
Arlington, Virginia 22209

by

Information Sciences Institute
University of Southern California
4676 Admiralty Way
Marina del Rey, California 90291

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

DoD STANDARD

TRANSMISSION CONTROL PROTOCOL

January 1980

Prepared for DARPA

by

Information Sciences Institute

University of Southern California

1400 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209

TABLE OF CONTENTS

PREFACE

This document describes the DoD Standard Transmission Control Protocol (TCP). There have been eight earlier editions of the ARPA TCP specification on which this standard is based, and the present text draws heavily from them. There have been many contributors to this work both in terms of concepts and in terms of text. This edition incorporates the addition of security, compartmentation, and precedence concepts into the TCP specification.

Jon Postel

Editor

January 1980
RFC:761
IEN:129
Replaces: IENs 124, 112,
81, 55, 44, 40, 27, 21, 5

DOD STANDARD

TRANSMISSION CONTROL PROTOCOL

1. INTRODUCTION

The Transmission Control Protocol (TCP) is intended for use as a highly reliable host-to-host protocol between hosts in packet-switched computer communication networks, and especially in interconnected systems of such networks.

This document describes the functions to be performed by the Transmission Control Protocol, the program that implements it, and its interface to programs or users that require its services.

1.1. Motivation

Computer communication systems are playing an increasingly important role in military, government, and civilian environments. This document primarily focuses its attention on military computer communication requirements, especially robustness in the presence of communication unreliability and availability in the presence of congestion, but many of these problems are found in the civilian and government sector as well.

As strategic and tactical computer communication networks are developed and deployed, it is essential to provide means of interconnecting them and to provide standard interprocess communication protocols which can support a broad range of applications. In anticipation of the need for such standards, the Deputy Undersecretary of Defense for Research and Engineering has declared the Transmission Control Protocol (TCP) described herein to be a basis for DoD-wide inter-process communication protocol standardization.

TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications. The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks. Very few assumptions are made as to the reliability of the communication protocols below the TCP layer. TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols. In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.

TCP is based on concepts first described by Cerf and Kahn in [1]. The TCP fits into a layered protocol architecture just above a basic Internet Protocol [2] which provides a way for the TCP to send and receive variable-length segments of information enclosed in internet datagram "envelopes". The internet datagram provides a means for addressing source and destination TCPs in different networks. The internet protocol also deals with any fragmentation or reassembly of the TCP segments required to achieve transport and delivery through multiple networks and interconnecting gateways. The internet protocol also carries information on the precedence, security classification and compartmentation of the TCP segments, so this information can be communicated end-to-end across multiple networks.

Protocol Layering

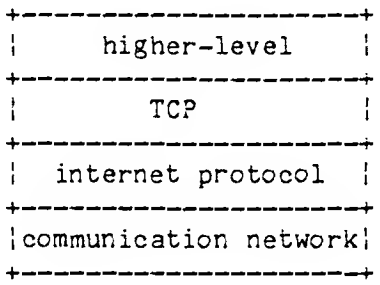


Figure 1

Much of this document is written in the context of TCP implementations which are co-resident with higher level protocols in the host computer. As a practical matter, many computer systems will be connected to networks via front-end computers which house the TCP and internet protocol layers, as well as network specific software. The TCP specification describes an interface to the higher level protocols which appears to be implementable even for the front-end case, as long as a suitable host-to-front end protocol is implemented.

1.2. Scope

The TCP is intended to provide a reliable process-to-process communication service in a multinetwork environment. The TCP is intended to be a host-to-host protocol in common use in multiple networks.

1.3. About this Document

This document represents a specification of the behavior required of any TCP implementation, both in its interactions with higher level protocols and in its interactions with other TCPs. The rest of this

section offers a very brief view of the protocol interfaces and operation. Section 2 summarizes the philosophical basis for the TCP design. Section 3 offers both a detailed description of the actions required of TCP when various events occur (arrival of new segments, user calls, errors, etc.) and the details of the formats of TCP segments.

1.4. Interfaces

The TCP interfaces on one side to user or application processes and on the other side to a lower level protocol such as Internet Protocol.

The interface between an application process and the TCP is illustrated in reasonable detail. This interface consists of a set of calls much like the calls an operating system provides to an application process for manipulating files. For example, there are calls to open and close connections and to send and receive letters on established connections. It is also expected that the TCP can asynchronously communicate with application programs. Although considerable freedom is permitted to TCP implementors to design interfaces which are appropriate to a particular operating system environment, a minimum functionality is required at the TCP/user interface for any valid implementation.

The interface between TCP and lower level protocol is essentially unspecified except that it is assumed there is a mechanism whereby the two levels can asynchronously pass information to each other. Typically, one expects the lower level protocol to specify this interface. TCP is designed to work in a very general environment of interconnected networks. The lower level protocol which is assumed throughout this document is the Internet Protocol [2].

1.5. Operation

As noted above, the primary purpose of the TCP is to provide reliable, securable logical circuit or connection service between pairs of processes. To provide this service on top of a less reliable internet communication system requires facilities in the following areas:

Basic Data Transfer
Reliability
Flow Control
Multiplexing
Connections
Precedence and Security

The basic operation of the TCP in each of these areas is described in the following paragraphs.

Basic Data Transfer:

The TCP is able to transfer a continuous stream of octets in each direction between its users by packaging some number of octets into segments for transmission through the internet system. In this stream mode, the TCPs decide when to block and forward data at their own convenience.

For users who desire a record-oriented service, the TCP also permits the user to submit records, called letters, for transmission. When the sending user indicates a record boundary (end-of-letter), this causes the TCPs to promptly forward and deliver data up to that point to the receiver.

Reliability:

The TCP must recover from data that is damaged, lost, duplicated, or delivered out of order by the internet communication system. This is achieved by assigning a sequence number to each octet transmitted, and requiring a positive acknowledgment (ACK) from the receiving TCP. If the ACK is not received within a timeout interval, the data is retransmitted. At the receiver, the sequence numbers are used to correctly order segments that may be received out of order and to eliminate duplicates. Damage is handled by adding a checksum to each segment transmitted, checking it at the receiver, and discarding damaged segments.

As long as the TCPs continue to function properly and the internet system does not become completely partitioned, no transmission errors will affect the users. TCP recovers from internet communication system errors.

Flow Control:

TCP provides a means for the receiver to govern the amount of data sent by the sender. This is achieved by returning a "window" with every ACK indicating a range of acceptable sequence numbers beyond the last segment successfully received. For stream mode, the window indicates an allowed number of octets that the sender may transmit before receiving further permission. For record mode, the window indicates an allowed amount of buffer space the sender may consume, this may be more than the number of data octets transmitted if there is a mismatch between letter size and buffer size.

2. PHILOSOPHY

2.1. Elements of the Internetwork System

The internetwork environment consists of hosts connected to networks which are in turn interconnected via gateways. It is assumed here that the networks may be either local networks (e.g., the ETHERNET) or large networks (e.g., the ARPANET), but in any case are based on packet switching technology. The active agents that produce and consume messages are processes. Various levels of protocols in the networks, the gateways, and the hosts support an interprocess communication system that provides two-way data flow on logical connections between process ports.

We specifically assume that data is transmitted from host to host through means of a set of networks. When we say network, we have in mind a packet switched network (PSN). This assumption is probably unnecessary, since a circuit switched network or a hybrid combination of the two could also be used; but for concreteness, we explicitly assume that the hosts are connected to one or more packet switches of a PSN.

The term packet is used generically here to mean the data of one transaction between a host and a packet switch. The format of data blocks exchanged between the packet switches in a network will generally not be of concern to us.

Hosts are computers attached to a network, and from the communication network's point of view, are the sources and destinations of packets. Processes are viewed as the active elements in host computers (in accordance with the fairly common definition of a process as a program in execution). Even terminals and files or other I/O devices are viewed as communicating with each other through the use of processes. Thus, all communication is viewed as inter-process communication.

Since a process may need to distinguish among several communication streams between itself and another process (or processes), we imagine that each process may have a number of ports through which it communicates with the ports of other processes.

2.2. Model of Operation

Processes transmit data by calling on the TCP and passing buffers of data as arguments. The TCP packages the data from these buffers into segments and calls on the internet module to transmit each segment to the destination TCP. The receiving TCP places the data from a segment into the receiving user's buffer and notifies the receiving user. The TCPs include control information in the segments which they use to ensure reliable ordered data transmission.

The model of internet communication is that there is an internet protocol module associated with each TCP which provides an interface to the local network. This internet module packages TCP segments inside internet datagrams and routes these datagrams to a destination internet module or intermediate gateway. To transmit the datagram through the local network, it is embedded in a local network packet.

The packet switches may perform further packaging, fragmentation, or other operations to achieve the delivery of the local packet to the destination internet module.

At a gateway between networks, the internet datagram is "unwrapped" from its local packet and examined to determine through which network the internet datagram should travel next. The internet datagram is then "wrapped" in a local packet suitable to the next network and routed to the next gateway, or to the final destination.

A gateway is permitted to break up an internet datagram into smaller internet datagram fragments if this is necessary for transmission through the next network. To do this, the gateway produces a set of internet datagrams; each carrying a fragment. Fragments may be broken into smaller ones at intermediate gateways. The internet datagram fragment format is designed so that the destination internet module can reassemble fragments into internet datagrams.

A destination internet module unwraps the segment from the datagram (after reassembling the datagram, if necessary) and passes it to the destination TCP.

This simple model of the operation glosses over many details. One important feature is the type of service. This provides information to the gateway (or internet module) to guide it in selecting the service parameters to be used in traversing the next network. Included in the type of service information is the precedence of the datagram. Datagrams may also carry security information to permit host and gateways that operate in multilevel secure environments to properly segregate datagrams for security considerations.

2.3. The Host Environment

The TCP is assumed to be a module in a time sharing operating system. The users access the TCP much like they would access the file system. The TCP may call on other operating system functions, for example, to manage data structures. The actual interface to the network is assumed to be controlled by a device driver module. The TCP does not call on the network device driver directly, but rather calls on the internet datagram protocol module which may in turn call on the device driver.

Though it is assumed here that processes are supported by the host operating system, the mechanisms of TCP do not preclude implementation of the TCP in a front-end processor. However, in such an implementation, a host-to-front-end protocol must provide the functionality to support the type of TCP-user interface described above.

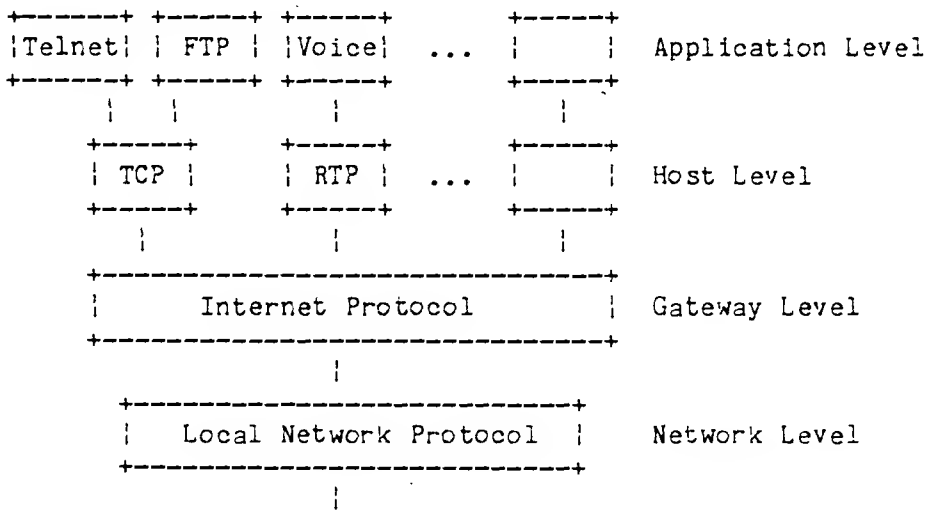
2.4. Interfaces

The TCP/user interface provides for calls made by the user on the TCP to OPEN or CLOSE a connection, to SEND or RECEIVE data, or to obtain STATUS about a connection. These calls are like other calls from user programs on the operating system, for example, the calls to open, read from, and close a file.

The TCP/internet interface provides calls to send and receive datagrams addressed to TCP modules in hosts anywhere in the internet system. These calls have parameters for passing the address, type of service, precedence, security, and other control information.

2.5. Relation to Other Protocols

The following diagram illustrates the place of the TCP in the protocol hierarchy:



Protocol Relationships

Figure 2.

It is expected that the TCP will be able to support higher level protocols efficiently. It should be easy to interface higher level protocols like the ARPANET Telnet [3] or AUTODIN II THP to the TCP.

2.6. Reliable Communication

A stream of data sent on a TCP connection is delivered reliably and in order at the destination.

Transmission is made reliable via the use of sequence numbers and acknowledgments. Conceptually, each octet of data is assigned a sequence number. The sequence number of the first octet of data in a segment is the sequence number transmitted with that segment and is called the segment sequence number. Segments also carry an acknowledgment number which is the sequence number of the next expected data octet of transmissions in the reverse direction. When the TCP transmits a segment, it puts a copy on a retransmission queue and starts a timer; when the acknowledgment for that data is received, the segment is deleted from the queue. If the acknowledgment is not received before the timer runs out, the segment is retransmitted.

An acknowledgment by TCP does not guarantee that the data has been delivered to the end user, but only that the receiving TCP has taken the responsibility to do so.

To govern the flow of data into a TCP, a flow control mechanism is employed. The the data receiving TCP reports a window to the sending TCP. This window specifies the number of octets, starting with the acknowledgment number that the data receiving TCP is currently prepared to receive.

2.7. Connection Establishment and Clearing

To identify the separate data streams that a TCP may handle, the TCP provides a port identifier. Since port identifiers are selected independently by each operating system, TCP, or user, they might not be unique. To provide for unique addresses at each TCP, we concatenate an internet address identifying the TCP with a port identifier to create a socket which will be unique throughout all networks connected together.

A connection is fully specified by the pair of sockets at the ends. A local socket may participate in many connections to different foreign sockets. A connection can be used to carry data in both directions, that is, it is "full duplex".

TCPs are free to associate ports with processes however they choose. However, several basic concepts seem necessary in any implementation.

There must be well-known sockets which the TCP associates only with the "appropriate" processes by some means. We envision that processes may "own" ports, and that processes can only initiate connections on the ports they own. (Means for implementing ownership is a local issue, but we envision a Request Port user command, or a method of uniquely allocating a group of ports to a given process, e.g., by associating the high order bits of a port name with a given process.)

A connection is specified in the OPEN call by the local port and foreign socket arguments. In return, the TCP supplies a (short) local connection name by which the user refers to the connection in subsequent calls. There are several things that must be remembered about a connection. To store this information we imagine that there is a data structure called a Transmission Control Block (TCB). One implementation strategy would have the local connection name be a pointer to the TCB for this connection. The OPEN call also specifies whether the connection establishment is to be actively pursued, or to be passively waited for.

A passive OPEN request means that the process wants to accept incoming connection requests rather than attempting to initiate a connection. Often the process requesting a passive OPEN will accept a connection request from any caller. In this case a foreign socket of all zeros is used to denote an unspecified socket. Unspecified foreign sockets are allowed only on passive OPENs.

A service process that wished to provide services for unknown other processes could issue a passive OPEN request with an unspecified foreign socket. Then a connection could be made with any process that requested a connection to this local socket. It would help if this local socket were known to be associated with this service.

Well-known sockets are a convenient mechanism for a priori associating a socket address with a standard service. For instance, the "Telnet-Server" process might be permanently assigned to a particular socket, and other sockets might be reserved for File Transfer, Remote Job Entry, Text Generator, Echoer, and Sink processes (the last three being for test purposes). A socket address might be reserved for access to a "Look-Up" service which would return the specific socket at which a newly created service would be provided. The concept of a well-known socket is part of the TCP specification, but the assignment of sockets to services is outside this specification.

Processes can issue passive OPENs and wait for matching calls from other processes and be informed by the TCP when connections have been established. Two processes which issue calls to each other at the same time are correctly connected. This flexibility is critical for

the support of distributed computing in which components act asynchronously with respect to each other.

There are two cases for matching the sockets in the local request and an incoming segment. In the first case, the local request has fully specified the foreign socket. In this case, the match must be exact. In the second case, the local request has left the foreign socket unspecified. In this case, any foreign socket is acceptable as long as the local sockets match.

If there are several pending passive OPENs (recorded in TCBs) with the same local socket, an incoming segment should be matched to a request with the specific foreign socket in the segment, if such a request exists, before selecting a request with an unspecified foreign socket.

The procedures to establish and clear connections utilize synchronize (SYN) and finis (FIN) control flags and involve an exchange of three messages. This exchange has been termed a three-way hand shake [4].

A connection is initiated by the rendezvous of an arriving segment containing a SYN and a waiting TCB entry created by a user OPEN command. The matching of local and foreign sockets determines when a connection has been initiated. The connection becomes "established" when sequence numbers have been synchronized in both directions.

The clearing of a connection also involves the exchange of segments, in this case carrying the FIN control flag.

2.8. Data Communication

The data that flows on a connection may be thought of as a stream of octets, or as a sequence of records. In TCP the records are called letters and are of variable length. The sending user indicates in each SEND call whether the data in that call completes a letter by the setting of the end-of-letter parameter.

The length of a letter may be such that it must be broken into segments before it can be transmitted to its destination. We assume that the segments will normally be reassembled into a letter before being passed to the receiving process. A segment may contain all or a part of a letter, but a segment never contains parts of more than one letter. The end of a letter is marked by the appearance of an EOL control flag in a segment. A sending TCP is allowed to collect data from the sending user and to send that data in segments at its own convenience, until the end of letter is signaled then it must send all unsent data. When a receiving TCP has a complete letter, it must not wait for more data from the sending TCP before passing the letter to the receiving process.

There is a coupling between letters as sent and the use of buffers of data that cross the TCP/user interface. Each time an end-of-letter (EOL) flag is associated with data placed into the receiving user's buffer, the buffer is returned to the user for processing even if the buffer is not filled. If a letter is longer than the user's buffer, the letter is passed to the user in buffer size units, the last of which may be only partly full. The receiving TCP's buffer size may be communicated to the sending TCP when the connection is being established.

The TCP is responsible for regulating the flow of segments on the connections, as a way of preventing itself from becoming saturated or overloaded with traffic. This is done using a window flow control mechanism. The data receiving TCP reports to the data sending TCP a window which is the range of sequence numbers of data octets that data receiving TCP is currently prepared to accept.

TCP also provides a means to communicate to the receiver of data that at some point further along in the data stream than the receiver is currently reading there is urgent data. TCP does not attempt to define what the user specifically does upon being notified of pending urgent data, but the general notion is that the receiving process should take action to read through the end urgent data quickly.

2.9. Precedence and Security

The TCP makes use of the internet protocol type of service field and security option to provide precedence and security on a per connection basis to TCP users. Not all TCP modules will necessarily function in a multilevel secure environment, some may be limited to unclassified use only, and others may operate at only one security level and compartment. Consequently, some TCP implementations and services to users may be limited to a subset of the multilevel secure case.

TCP modules which operate in a multilevel secure environment should properly mark outgoing segments with the security, compartment, and precedence. Such TCP modules should also provide to their users or higher level protocols such as Telnet or THP an interface to allow them to specify the desired security level, compartment, and precedence of connections.

2.10. Robustness Principle

TCP implementations should follow a general principle of robustness: be conservative in what you do, be liberal in what you accept from others.

GLOSSARY

1822

BBN Report 1822, "The Specification of the Interconnection of a Host and an IMP". The specification of interface between a host and the ARPANET.

ACK

A control bit (acknowledge) occupying no sequence space, which indicates that the acknowledgment field of this segment specifies the next sequence number the sender of this segment is expecting to receive, hence acknowledging receipt of all previous sequence numbers.

ARPANET message

The unit of transmission between a host and an IMP in the ARPANET. The maximum size is about 1012 octets (8096 bits).

ARPANET packet

A unit of transmission used internally in the ARPANET between IMPs. The maximum size is about 126 octets (1008 bits).

buffer size

An option (buffer size) used to state the receive data buffer size of the sender of this option. May only be sent in a segment that also carries a SYN.

connection

A logical communication path identified by a pair of sockets.

datagram

A message sent in a packet switched computer communications network.

Destination Address

The destination address, usually the network and host identifiers.

EOL

A control bit (End of Letter) occupying no sequence space, indicating that this segment ends a logical letter with the last data octet in the segment. If this end of letter causes a less than full buffer to be released to the user and the connection buffer size is not one octet then the end-of-letter/buffer-size adjustment to the receive sequence number must be made.

FIN

A control bit (finis) occupying one sequence number, which indicates that the sender will send no more data or control occupying sequence space.

fragment

A portion of a logical unit of data, in particular an internet fragment is a portion of an internet datagram.

FTP

A file transfer protocol.

header

Control information at the beginning of a message, segment, fragment, packet or block of data.

host

A computer. In particular a source or destination of messages from the point of view of the communication network.

Identification

An Internet Protocol field. This identifying value assigned by the sender aids in assembling the fragments of a datagram.

IMP

The Interface Message Processor, the packet switch of the ARPANET.

internet address

A source or destination address specific to the host level.

internet datagram

The unit of data exchanged between an internet module and the higher level protocol together with the internet header.

internet fragment

A portion of the data of an internet datagram with an internet header.

IP

Internet Protocol.

IRS

The Initial Receive Sequence number. The first sequence number used by the sender on a connection.

ISN

The Initial Sequence Number. The first sequence number used on a connection, (either ISS or IRS). Selected on a clock based procedure.

ISS

The Initial Send Sequence number. The first sequence number used by the sender on a connection.

leader

Control information at the beginning of a message or block of data. In particular, in the ARPANET, the control information on an ARPANET message at the host-IMP interface.

left sequence

This is the next sequence number to be acknowledged by the data receiving TCP (or the lowest currently unacknowledged sequence number) and is sometimes referred to as the left edge of the send window.

letter

A logical unit of data, in particular the logical unit of data transmitted between processes via TCP.

local packet

The unit of transmission within a local network.

module

An implementation, usually in software, of a protocol or other procedure.

MSL

Maximum Segment Lifetime, the time a TCP segment can exist in the internetwork system. Arbitrarily defined to be 2 minutes.

octet

An eight bit byte.

Options

An Option field may contain several options, and each option may be several octets in length. The options are used primarily in testing situations; for example, to carry timestamps. Both the Internet Protocol and TCP provide for options fields.

packet

A package of data with a header which may or may not be

logically complete. More often a physical packaging than a logical packaging of data.

port
: The portion of a socket that specifies which logical input or output channel of a process is associated with the data.

process
: A program in execution. A source or destination of data from the point of view of the TCP or other host-to-host protocol.

PSN
: A Packet Switched Network. For example, the ARPANET.

RCV.BS
: receive buffer size, the remote buffer size

RCV.LBB
: receive last buffer beginning

RCV.NXT
: receive next sequence number

RCV.UP
: receive urgent pointer

RCV.WND
: receive window

receive last buffer beginning
: This is the sequence number of the first octet of the most recent buffer. This value is use in calculating the next sequence number when a segment contains an end of letter indication.

receive next sequence number
: This is the next sequence number the local TCP is expecting to receive.

receive window
: This represents the sequence numbers the local (receiving) TCP is willing to receive. Thus, the local TCP considers that segments overlapping the range RCV.NXT to RCV.NXT + RCV.WND - 1 carry acceptable data or control. Segments containing sequence numbers entirely outside of this range are considered duplicates and discarded.

Multiplexing:

To allow for many processes within a single Host to use TCP communication facilities simultaneously, the TCP provides a set of addresses or ports within each host. Concatenated with the network and host addresses from the internet communication layer, this forms a socket. A pair of sockets uniquely identifies each connection. That is, a socket may be simultaneously used in multiple connections.

The binding of ports to processes is handled independently by each Host. However, it proves useful to attach frequently used processes (e.g., a "logger" or timesharing service) to fixed sockets which are made known to the public. These services can then be accessed through the known addresses. Establishing and learning the port addresses of other processes may involve more dynamic mechanisms.

Connections:

The reliability and flow control mechanisms described above require that TCPs initialize and maintain certain status information for each data stream. The combination of this information, including sockets, sequence numbers, and window sizes, is called a connection. Each connection is uniquely specified by a pair of sockets identifying its two sides.

When two processes wish to communicate, their TCP's must first establish a connection (initialize the status information on each side). When their communication is complete, the connection is terminated or closed to free the resources for other uses.

Since connections must be established between unreliable hosts and over the unreliable internet communication system, a handshake mechanism with clock-based sequence numbers is used to avoid erroneous initialization of connections.

Precedence and Security:

The users of TCP may indicate the security and precedence of their communication. Provision is made for default values to be used when these features are not needed.

RST

A control bit (reset), occupying no sequence space, indicating that the receiver should delete the connection without further interaction. The receiver can determine, based on the sequence number and acknowledgment fields of the incoming segment, whether it should honor the reset command or ignore it. In no case does receipt of a segment containing RST give rise to a RST in response.

RTP

Real Time Protocol: A host-to-host protocol for communication of time critical information.

Rubber EOL

An end of letter (EOL) requiring a sequence number adjustment to align the beginning of the next letter on a buffer boundary.

SEG.ACK

segment acknowledgment

SEG.LEN

segment length

SEG.PRC

segment precedence value

SEG.SEQ

segment sequence

SEG.UP

segment urgent pointer field

SEG.WND

segment window field

segment

A logical unit of data, in particular a TCP segment is the unit of data transfered between a pair of TCP modules.

segment acknowledgment

The sequence number in the acknowledgment field of the arriving segment.

segment length

The amount of sequence number space occupied by a segment, including any controls which occupy sequence space.

segment sequence
: The number in the sequence field of the arriving segment.

send last buffer beginning
: This is the sequence number of the first octet of the most recent buffer. This value is used in calculating the next sequence number when a segment contains an end of letter indication.

send sequence
: This is the next sequence number the local (sending) TCP will use on the connection. It is initially selected from an initial sequence number curve (ISN) and is incremented for each octet of data or sequenced control transmitted.

send window
: This represents the sequence numbers which the remote (receiving) TCP is willing to receive. It is the value of the window field specified in segments from the remote (data receiving) TCP. The range of sequence numbers which may be emitted by a TCP lies between SND.NXT and SND.UNA + SND.WND - 1.

SND.BS
: send buffer size, the local buffer size

SND.LBB
: send last buffer beginning

SND.NXT
: send sequence

SND.UNA
: left sequence

SND.UP
: send urgent pointer

SND.WL
: send sequence number at last window update

SND.WND
: send window

socket
: An address which specifically includes a port identifier, that is, the concatenation of an Internet Address with a TCP port.

Source Address
: The source address, usually the network and host identifiers.

SYN
: A control bit in the incoming segment, occupying one sequence number, used at the initiation of a connection, to indicate where the sequence numbering will start.

TCB
: Transmission control block, the data structure that records the state of a connection.

TCB.PRC
: The precedence of the connection.

TCP
: Transmission Control Protocol: A host-to-host protocol for reliable communication in internetwork environments.

TOS
: Type of Service, an Internet Protocol field.

Type of Service
: An Internet Protocol field which indicates the type of service for this internet fragment.

URG
: A control bit (urgent), occupying no sequence space, used to indicate that the receiving user should be notified to do urgent processing as long as there is data to be consumed with sequence numbers less than the value indicated in the urgent pointer.

urgent pointer
: A control field meaningful only when the URG bit is on. This field communicates the value of the urgent pointer which indicates the data octet associated with the sending user's urgent call.

REFERENCES

[1] Cerf, V., and R. Kahn, "A Protocol for Packet Network Intercommunication," IEEE Transactions on Communications, Vol. COM-22, No. 5, pp 637-648, May 1974.

[2] Postel, J. (ed.), "DOD Standard Internet Protocol," Defense Advanced Research Projects Agency, Information Processing Techniques Office, RFC 760, IEN 128, January 1980.

[3] Feinler, E. and J. Postel, ARPANET Protocol Handbook, Network Information Center, SRI International, Menlo Park, CA, January 1978.

[4] Dalal, Y. and C. Sunshine, "Connection Management in Transport Protocols," Computer Networks, Vol. 2, No. 6, pp. 454-473, December 1978.